Distributed Stable Matching with Similar Preference Lists

Pankaj Khanchandani[1] and Roger Wattenhofer[1]

1 **ETH Zurich, Switzerland**
`kpankaj@ethz.ch` (contact author), `wattenhofer@ethz.ch`

Abstract

Consider a complete bipartite graph of $2n$ nodes with n nodes on each side. In a round, each node can either send at most one message to a neighbor or receive at most one message from a neighbor. Each node has a preference list that ranks all its neighbors in a strict order from 1 to n. We introduce a non-negative similarity parameter $\Delta < n$ for the preference lists of nodes on one side only. For $\Delta = 0$, these preference lists are same and for $\Delta = n - 1$, they can be completely arbitrary. There is no restriction on the preference lists of the other side. We show that each node can compute its partner in a stable matching by receiving $O(n(\Delta + 1))$ messages of size $O(\log n)$ each. We also show that this is optimal (up to a logarithmic factor) if Δ is constant.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases distributed stable matching, similar preference lists, stable matching, stable marriage

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

In 1962, Gale and Shapley proposed an algorithm [5] to find a *stable matching* between a set of n men and n women, where each participant has a preference list ranking all the participants of opposite gender in a strict order. The algorithm computes a match for each man (and each woman) so that any pair of man and woman who are not matched do not both prefer each other to their current matches. The Gale-Shapley algorithm is one of the most influential results in 20th century science, and forms the basis of the 2012 Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel.

The Gale-Shapley algorithm is also one of the very first distributed algorithms, as all the unmatched men can send their proposals simultaneously. Despite its distributed design, there are worst-case problem instances which basically execute sequentially as there are $\Theta(n^2)$ steps with only a single unmatched man [8]. These worst-case problem instances however feel very contrived, as participants need completely divergent preferences. One might expect that real world preference lists will be somewhat similar.

In this paper, we want to know whether similar preference lists allow for designing faster distributed algorithms. Towards this, we introduce a parameter Δ which is the smallest number so that the difference between the maximum and minimum ranks assigned to any man is at most Δ. A smaller value of Δ implies that the difference in the positions of a man in the preference lists of any two women is also smaller and the preference lists of women are relatively similar. The preference lists of men are still allowed to be arbitrary, independent of Δ.

Each participant is represented by a node in our distributed model and computation proceeds in synchronous rounds. If we use the traditional message passing model of distributed computing, a man can send his proposals to all the n women simultaneously, where n can be arbitrarily large. This does not seem to be a realistic social construct. Instead, in this paper we use a communication model where each node can either send *or* receive at most one message in a single round. It is an exciting open question whether a node can compute its stable partner by receiving $o(n^2)$ values or messages of size $O(\log n)$ each.

We give an algorithm so that any node computes its partner in a stable matching by receiving $O(n(\Delta+1))$ messages of size $O(\log n)$ each. We also show that any node requires $\Omega(n/\log n)$ messages to compute its partner. Thus, our algorithm is nearly optimal for constant Δ.

2 Related Work

The seminal paper by Gale and Shapley [5] has inspired lots of research on stable matching. The book of Gusfield and Irving [6] extensively survey the problem and its many variants. Irving et al. give a version where the preference lists of one set of participants (say women) are derived from a master preference list so that the men present in the master list occur in the preference lists of women in the same order as in the master list [7]. This results in similar preference lists but the order of a group of men remains same across all the women. The similarity measure Δ that we use does not enforce a strict order on any particular group of men. To the best of our knowledge, this measure has not been discussed before.

Distributed stable matching with a goal of keeping the preference lists private to an individual has also been studied [2, 3]. In our work, our focus is to solve the problem rather collaboratively, minimizing the information required by each node to compute its stable partner. Kipnis and Patt-Shamir consider a distributed version in which a man is not required to rank all the women and give a lower bound on the information required by a node using an incomplete bipartite graph [9]. To understand the hardness of the problem when topology is not a limiting factor, we consider that each man ranks all the women and vice-versa or a complete bipartite graph.

Amira et al. consider a variant where a complete bipartite graph has weighted edges [1]. The preference list of a node is then derived according the order of weights of edges that are incident at the node. They give an algorithm in which any node can compute its partner using only $O(n\sqrt{n})$ values. For the problem instances that they consider, the preference lists of women is dependent on the preference lists of men or the other way around. In our case, the preference lists of men can be completely arbitrary and that of women can be chosen independently, only subject to the parameter Δ. Also, the values required by a node to compute a stable matching can be larger or smaller depending on the parameter Δ.

In parallel algorithms with $\Theta(n)$ processes, a process might still need to access $\Omega(n^2)$ values before a stable partner is computed [12, 13]. Distributed algorithms for approximate stable matching have also been studied. For example, Floréen et al. consider the case where the preference lists have bounded length and give a local algorithm for a stable matching with only few unstable edges [4]. Ostrovsky et al. give an algorithm for almost stable matching that terminates in poly-logarithmic rounds for arbitrary preferences [11].

3 Model

Let $\mathbb{M} = \{m_1, m_2, \ldots, m_n\}$ be the set of men and $\mathbb{W} = \{w_1, w_2, \ldots, w_n\}$ be the set of women. In a *matching*, any man is matched to at most one woman and any woman is matched to at most one man. A *perfect matching* is a matching where all the men and women are matched.

Given a perfect matching, a pair of man and woman that are not matched to each other is called a *blocking pair* if they prefer each other to their matched partners. Thus, a stable matching is a perfect matching without a blocking pair.

We denote the preference lists of men by matrix M where an entry M_{ij} is the j^{th} ranked woman by man m_i. Similarly, matrix W is so that an entry W_{ij} is the j^{th} ranked man by woman w_i. Thus, a woman w_i prefers man W_{ij} to the man W_{ik} if $j < k$ and a man m_i prefers woman M_{ij} to the woman M_{ik} if $j < k$. The similarity parameter Δ is the smallest value so that the following holds for the matrix W.

$$W_{ij} = W_{pq} \Rightarrow |j - q| \leq \Delta \tag{1}$$

Thus, the range in which a man is ranked by all the women is at most Δ large.

In the distributed setting, the set $\mathbb{M}$ and $\mathbb{W}$ are the nodes and the network is a complete bipartite graph with $\mathbb{M}$ and $\mathbb{W}$ on the opposite sides. The algorithm proceeds in sequence of synchronous rounds. In each round, a node can either

1. Send a message to at most one neighbor and do local computations; or
2. Receive a message from at most one neighbor and do local computations.

If several messages are sent to the same node in the same round, then none are received by the intended recipient. The messages are of size $O(\log n)$.

Let us first see that any algorithm for finding the stable matching in this model requires $\Omega(n/\log n)$ rounds when $\Delta = 0$. Afterwards, we will see a sequential algorithm and its analysis which we later develop into the distributed algorithm. Finally, we show its correctness.

4 Lower Bound

Figure 1 shows sample matrices M and W for $n = 5$ and $\Delta = 0$. As $\Delta = 0$, the preference lists of all the women are same. For simplicity, we assume that the man m_i is ranked i^{th} by all women.

$$M = \begin{pmatrix} w_1 & w_2 & w_3 & w_4 & w_5 \\ w_1 & w_4 & w_5 & w_3 & w_2 \\ w_4 & w_3 & w_1 & w_5 & w_2 \\ w_5 & w_4 & w_1 & w_2 & w_3 \\ w_5 & w_1 & w_2 & w_3 & w_4 \end{pmatrix} \qquad W = \begin{pmatrix} m_1 & m_2 & m_3 & m_4 & m_5 \\ m_1 & m_2 & m_3 & m_4 & m_5 \\ m_1 & m_2 & m_3 & m_4 & m_5 \\ m_1 & m_2 & m_3 & m_4 & m_5 \\ m_1 & m_2 & m_3 & m_4 & m_5 \end{pmatrix}$$

Figure 1 Sample Preference Matrices for $n = 5$, $\Delta = 0$

The man m_1 is ranked first by all the women. Thus, m_1 must be matched to his first preference, w_1, in any stable matching. Otherwise, m_1 and w_1 would form a blocking pair.

The man m_2 is ranked second by all the women and his most preferred woman is w_1. However, m_2 cannot be matched to w_1 as she must be matched to m_1. On the other hand, m_2 must be matched to w_4, which is his most preferred woman excluding w_1. Otherwise, m_2 and w_4 would form a blocking pair. We can generalize this into the following lemma.

▶ **Lemma 1.** *Consider the stable matching problem with n men and n women where any entry $W_{ij} = m_j$, i.e., $\Delta = 0$. In any stable matching, a man m_j is matched to his most preferred woman excluding the women matched to men m_i, $i < j$.*

Proof. Assume a stable matching S in which a j exists so that m_j is *not* matched to his most preferred women when the women matched to m_i, $i < j$ are excluded.

Say, that m_j is matched to a woman w and w^* is his most preferred woman excluding the women matched to m_i, $i < j$. Clearly, m_j prefers w^* to w. Consider the woman w^*. She is not matched to m_j by our assumption. Also, she is not matched to m_i for $i < j$ either. Thus, w^* prefers m_j to her match m_k, $k > j$. Hence, m_j and w^* form a blocking pair and S is not a stable matching. ◀

Thus, m_j should know the women matched to men m_i, $i < j$ to determine his partner. We exploit this to show a lower bound on the number of rounds necessary to compute the stable matching as follows. We will use the notation $[1, n]$ for the set of integers between 1 and n inclusive.

▶ **Lemma 2.** *It takes $\Omega(n/\log n)$ rounds to compute a stable matching for n men and n women when $W_{ij} = m_j$, i.e., $\Delta = 0$.*

Proof. Consider a man m_k, $k \in [1, n]$. As a corollary of Lemma 1, m_k is matched to a woman M_{kj}, $j \leq k$. Let B be the set of $k-1$ highest ranked women by m_k, i.e., $B = \{M_{kj} : j \in [1, k-1]\}$. Let A be the set of women matched to the men ranked higher than m_k by the women, i.e., $A = \{w_i : w_i \text{ matched to } m_j,\ j < k\}$. Using Lemma 1, $A = B$ if m_k is matched to M_{kk}. On the other hand, if m_k is not matched to M_{kk}, then m_k is matched to a woman in set B that is not in set A and $A \neq B$.

It is known that $\Omega(n)$ bits must be exchanged between two parties, if each of them is given a subset of $[1, n]$ and they want to determine if the subsets are same [10] [1]. Here, node m_k having set B determined if it was equal to set A present in the other nodes. Thus, $\Omega(n)$ bits must be exchanged with m_k. As the message size is $O(\log n)$, the algorithm must run for $\Omega(n/\log n)$ rounds. ◀

Let us now see a sequential algorithm in which any man needs to propose $O(\Delta)$ times to find the stable matching. Later, we will implement this algorithm in our distributed model.

5 Sequential Algorithm

We will use the notation $c(m_j)$ to denote the smallest column number of W in which the man m_j occurs. We show sample input matrices M and W for $n = 5$ and $\Delta = 2$ in Figure 2. For this input, $c(m_5) = c(m_3) = 1$, $c(m_2) = 2$, $c(m_4) = 3$ and $c(m_1) = 4$. Also, we define a bijective function $l : [1, n] \to [1, n]$ so that $c(m_j) < c(m_k) \Rightarrow l(j) < l(k)$. As the function l is bijective, l^{-1} is defined as well. For example, following is an assignment of l and l^{-1} given the preference matrix W in Figure 2: $l(5) = 1$, $l(3) = 2$, $l(2) = 3$, $l(4) = 4$, $l(1) = 5$ and $l^{-1}(1) = 5$, $l^{-1}(2) = 3$, $l^{-1}(3) = 2$, $l^{-1}(4) = 4$, $l^{-1}(5) = 1$.

$$M = \begin{pmatrix} w_1 & w_5 & w_4 & w_3 & w_2 \\ w_4 & w_3 & w_1 & w_5 & w_2 \\ w_1 & w_4 & w_5 & w_3 & w_2 \\ w_5 & w_4 & w_2 & w_1 & w_3 \\ w_1 & w_2 & w_3 & w_4 & w_5 \end{pmatrix} \qquad W = \begin{pmatrix} m_5 & m_3 & m_2 & m_4 & m_1 \\ m_3 & m_2 & m_5 & m_1 & m_4 \\ m_3 & m_5 & m_4 & m_2 & m_1 \\ m_5 & m_2 & m_3 & m_4 & m_1 \\ m_5 & m_3 & m_2 & m_1 & m_4 \end{pmatrix}$$

Figure 2 Sample Preference Matrices for $n = 5$, $\Delta = 2$

[1] The result also applies if the sizes of the subsets are same as in our case. As otherwise, one could exchange the sizes in $O(\log n)$ bits and compute the result using $o(n)$ bits.

Algorithm 1 describes the sequential stable matching algorithm. The set F contains men that are free and is initialized with all the men. The variable i is the maximum column number so that there is a man m_j with $c(m_j) = i$ that already proposed to someone. The free man that has the smallest value assigned by function l proposes the next women available in his list. Not all are available as some of them are deleted from his list during the algorithm. A woman accepts a proposal if she is free or if the proposal is better than her current partner. If it was the first proposal she accepted, then she is deleted from the preference lists of all the men m_j that have $c(m_j) > i + \Delta$.

Algorithm 1: Sequential Stable Matching: The loop at Line 3 ends after all men are matched. If a woman w receives her first proposal (Line 6), then w is deleted from the preference list of all men m_j where $c(m_j) > i + \Delta$ (loop at Line 14).

```
1  $F \leftarrow \mathbb{M}$;
2  $i \leftarrow 1$;
3  while $F \neq \emptyset$ do
4      $m \leftarrow m_j$ where $j = l^{-1}(\min\{l(k) : m_k \in F\})$;
5      $i = \max\{i, c(m)\}$;
6      $m$ proposes to the next available (not already deleted) woman $w$ from his
         preference list;
7      if $w$ is enaged and does not prefer $m$ to her current partner then
8          $w$ rejects $m$'s proposal;
9      else
10         if $w$ is engaged to a less preferred partner $m'$ then
11             $w$ disengages with $m'$;
12             $F \leftarrow F \cup \{m'\}$;
13         else // $w$ is not engaged
14             forall $j \in [1, n]$ do
15                 if $c(m_j) > i + \Delta$ then
16                     Delete $w$ from $m_j$'s preference list;
17         $w$ accepts $m$;
18         $F \leftarrow F \backslash \{m\}$;
```

We show an execution of the algorithm in Table 1. It can be checked that the final matching is indeed a stable matching. Also note that no more than three proposals $(\Delta + 1)$ are required for any man m_j. Let us now analyze the algorithm to see if we can generalize these observations.

6 Analysis of the Sequential Algorithm

It is rather easy to show that the algorithm always maintains a matching, i.e., each node is matched to at most one neighbor.

▶ **Lemma 3.** *Algorithm 1 always maintains a matching.*

Proof. A woman is never matched to more than one man as whenever she accepts a proposal, she always disengages with her current partner (if any). A man sends a proposal only if he is in set F. Initially, F contains all the men which are free. Whenever a man is accepted, he

F	Propose	i	Delete	Matching
$\mathbb{M}$	$m_5 \to w_1$	1	M_{11}	$\{(m_5, w_1)\}$
$\mathbb{M} \backslash m_5$	$m_3 \to w_1$	1		$\{(m_5, w_1)\}$
$\mathbb{M} \backslash m_5$	$m_3 \to w_4$	1	M_{13}	$\{(m_5, w_1), (m_3, w_4)\}$
$\mathbb{M} \backslash \{m_5, m_3\}$	$m_2 \to w_4$	2		$\{(m_5, w_1), (m_2, w_4)\}$
$\mathbb{M} \backslash \{m_5, m_2\}$	$m_3 \to w_5$	2		$\{(m_5, w_1), (m_2, w_4), (m_3, w_5)\}$
$\{m_1, m_4\}$	$m_4 \to w_5$	3		$\{(m_5, w_1), (m_2, w_4), (m_3, w_5)\}$
$\{m_1, m_4\}$	$m_4 \to w_4$	3		$\{(m_5, w_1), (m_2, w_4), (m_3, w_5)\}$
$\{m_1, m_4\}$	$m_4 \to w_2$	3		$\{(m_5, w_1), (m_2, w_4), (m_3, w_5), (m_4, w_2)\}$
$\{m_1\}$	$m_1 \to w_5$	4		$\{(m_5, w_1), (m_2, w_4), (m_3, w_5), (m_4, w_2)\}$
$\{m_1\}$	$m_1 \to w_3$	4		$\{(m_5, w_1), (m_2, w_4), (m_3, w_5), (m_4, w_2), (m_1, w_3)\}$

Table 1 An execution of Algorithm 1 on preference matrices of Figure 2. We use the following definition of function l: $l(5) = 1$, $l(3) = 2$, $l(2) = 3$, $l(4) = 4$ and $l(1) = 5$. Thus, we choose a free man in Line 4 that occurs earliest in the following list: m_5, m_3, m_2, m_4, m_1. We show the change in variables during the subsequent iterations of the loop at Line 3. The column 'F' shows the value of the variable F at the start of the iteration. The column 'Propose' is the proposal made during the iteration. The column i shows the value of variable i just before entries from preference lists are deleted. The column 'Delete' shows entries of M deleted during the iteration. The column 'Matching' is the matching at the end of the iteration.

is removed from F and whenever a man is disengaged, he is added to F. Thus, a matched man never sends a proposal and a man is matched to at most one woman. ◀

However, it is not clear that the algorithm computes a perfect matching as the loop of Line 3 may not terminate. We show that this does not happen using contradiction as follows.

▶ **Lemma 4.** *Algorithm 1 computes a perfect matching.*

Proof. If the algorithm terminates, then none of the men are free. Using Lemma 3 the statement holds. Thus, we now only need to show that the algorithm terminates.

Assume that the algorithm does not terminate. Then, there is a man m that has exhausted his preference list and is unmatched. Now, there are two possibilities for each woman w in m's preference list. First: m proposed to w and got rejected or disengaged later. Second: m did not propose to w as she was already deleted from m's preference list. In either case, w would still be matched as w never rejects a proposal if she is free and only accepts better proposals if already matched. This means that n women are matched to at most $n-1$ other men (excluding m). Using Lemma 3, this is a contradiction. ◀

And now that we have a perfect matching upon termination, we can assume a blocking pair in that perfect matching and show that such a pair does not exist. We introduce the notation $W_{[a,b][x,y]}$ to denote the set of elements in the sub-matrix of W formed between rows a, b inclusive and columns x, y inclusive. We will also use the shorthand $*$ to denote the range $[1, n]$ and a to denote the single element range $[a, a]$.

▶ **Lemma 5.** *The matching computed by Algorithm 1 is a stable matching.*

Proof. We know from Lemma 4 that the algorithm terminates in a perfect matching. Assume that m and w' form a blocking pair and w' prefers m to her final match m'.

Now, either m did or did not propose to w'. If m proposed to w', then it got rejected or disengaged later and w' prefers m' to m, a contradiction. If m did not propose to w', then

w' was removed from m's preference list. Say, w' was matched to m'' and $i = p$ when she was removed. Then, we conclude that $c(m) > p + \Delta$. Also, $c(m'') \leq p$ as m'' is matched to w' when $i = p$. Consequently, $m'' \notin W_{*[p+\Delta+1,n]}$ due to (1). Thus, w' prefers m'' to m. As w' only accepts better proposals in the future and m'' was her first match, she also prefers m' to m, a contradiction. ◀

There is an interesting property of this algorithm, namely, no man proposes more than $3\Delta + 1$ times. We will first show a bound on the number of men up to a certain number of columns of W. Later, we use this bound to show this property.

▶ **Lemma 6.** *The total number of men in $W_{*[1,j]}$ is at least j and at most $j + \Delta$.*

Proof. The elements of any given row $i \in [1, n]$ of W are all different and thus they are different in a given range $[1, j]$. Thus, we have $|W_{i[1,j]}| = j$ and $|W_{*[1,j]}| \geq j$.

If a man $m \in W_{*[1,j]}$, then (1) implies that $m \notin W_{*[j+\Delta+1,n]}$. As m occurs in every row of W, $m \in W_{i[1,j+\Delta]}$ for $i \in [1, n]$. Thus, $W_{*[1,j]} \subseteq W_{i[1,j+\Delta]}$ for $i \in [1, n]$ and we have $|W_{*[1,j]}| \leq (j + \Delta) - 1 + 1 = j + \Delta$. ◀

Now, we can use the above lemma to bound the number of proposals that are made by a man.

▶ **Lemma 7.** *The maximum number of proposals made by a man in Algorithm 1 is $3\Delta + 1$.*

Proof. Let m be some man and say that $c(m) = p$. Consider the first time t when all the men m_j with $c(m_j) \leq p - \Delta - 1$ are matched. When $i > p - \Delta - 1$ for the first time, then all the men m_j with $c(m_j) \leq p - \Delta - 1$ are already matched. Thus, $i \leq p - \Delta - 1$ at time t. Note that the set of men m_j with $c(m_j) \leq p - \Delta - 1$ is same as $W_{*[1,p-\Delta-1]}$. Using Lemma 6, this set is at least $p - \Delta - 1$ in size. Thus, at least $p - \Delta - 1$ free women received proposals when $i \leq p - \Delta - 1$. As $c(m) = p > (p - \Delta - 1) + \Delta$, at least $p - \Delta - 1$ women were deleted from m's preference list.

At time t, let $D(m)$ be the set of men matched to the women deleted from m's preference list. As $i \leq p - \Delta - 1$ at time t,

$$m' \in D(m) \Rightarrow m' \in W_{*[1,p-\Delta-1]} \,. \tag{2}$$

Let $R(m)$ be the set of men that are ranked better than m at least once by some woman. Thus, we have $D(m) \subseteq R(m)$. Say that m makes x proposals in total. Then, m was rejected or disengaged $x - 1$ times due to a man from $R(m)$. As a woman only accepts better proposals in future, there are at least $|D(m)| + (x - 1)$ women that are matched to men in $R(m)$ when algorithm terminates. Thus,

$$\begin{aligned}
|D(m)| + (x-1) &\leq |R(m)| \\
(p - \Delta - 1) + (x - 1) &\leq |R(m)| && ((2)) \\
p - \Delta + x - 2 &\leq |W_{*[1,p+\Delta]} \backslash m| && ((1)) \\
p - \Delta + x - 2 &\leq |W_{*[1,p+\Delta]}| - 1 && (c(m) = p) \\
p - \Delta + x - 2 &\leq p + 2\Delta - 1 && (\text{Lemma 6}) \\
x &\leq 3\Delta + 1 \,.
\end{aligned}$$

◀

Let us now use our analysis to build a distributed version of the algorithm.

7 Distributed Algorithm

Before describing the distributed algorithm, we will compute some quantities that will give us some useful abstractions. Let us first compute Δ at every node in $\mathbb{M}$. We let $r(w_i, m)$ to be the rank assigned by w_i to $m \in \mathbb{M}$, i.e., $W_{ip} = m$ for $p = r(w_i, m)$.

▶ **Lemma 8.** *All nodes $m_j \in \mathbb{M}$ know $r(w_i, m_j)$ for all $i \in [1, n]$ after the following procedure is repeated for $p \in [0, n-1]$: for $k \in [1, n]$, w_k sends $r(w_k, m_{(k+p) \bmod n+1})$ to $m_{(k+p) \bmod n+1}$ in round p.*

Proof. Consider the nodes w_a and w_b where $a \neq b$. In any given round p the expression $(a+p) \bmod n+1 \neq (b+p) \bmod n+1$. Thus, messages sent by w_a and w_b in round p are sent to different nodes and received by them consequently. Also, the expression $(a+p) \bmod n+1$ covers every value in the range $[1, n]$ for a given a when p covers the range $[0, n-1]$. Thus, all nodes $w_a \in \mathbb{W}$ send $r(w_a, m_k)$ for $k \in [1, n]$. ◀

Now, using the previous lemma each node $m_j \in \mathbb{M}$ can compute the value of Δ.

▶ **Lemma 9.** *In $O(n)$ rounds, each node $m_j \in \mathbb{M}$ can compute the value of Δ.*

Proof. Using Lemma 8, any node m_j can compute the set $R_j = \{r(w_i, m_j) : i \in [1, n]\}$. Thus, m_j can compute $R_j^{max} = \max R_j$, $R_j^{min} = \min R_j$ and $\Delta_j = R_j^{max} - R_j^{min}$. Now in n rounds, each node m_j can send Δ_j to w_1. Then, w_1 can compute $\Delta = \max\{\Delta_j : j \in [1, n]\}$ and send it to all nodes m_j in another n rounds. ◀

Let us also compute the function c at the nodes $\mathbb{M}$, i.e., each node $m \in \mathbb{M}$ computes the set $\{c(m_j) : j \in [1, n]\}$. This function can be used to compute functions such as l locally that would be useful for distributed algorithm design.

▶ **Lemma 10.** *In $O(n)$ rounds, each node $m_j \in \mathbb{M}$ can compute the function c.*

Proof. Using Lemma 8, each node $m_j \in \mathbb{M}$ can compute the set $R_j = \{r(w_i, m_j) : i \in [1, n]\}$. Thus, m_j can compute $c(m_j) = \min R_j$.

We define a circular path P of nodes from $\mathbb{M}$ so that the next node of a node m_j is $m_{j \bmod n+1}$. Each node m_j already knows $v_j = \langle m_j, c(m_j)\rangle$. Now, in 2 rounds each node $m_j \in P$ can send v_j to next node on P by forwarding it through $w_{j \bmod n+1}$. If this is done $n-1$ times, each value v_j reaches all the other $n-1$ nodes on path P and each node in $\mathbb{M}$ stores the complete function c. ◀

▶ **Corollary 11.** *In $O(n)$ rounds, each node $m_j \in \mathbb{M}$ can compute the function l.*

Proof. Using Lemma 10, each node $m_j \in \mathbb{M}$ computes the function c in $O(n)$ rounds. Now, l can computed locally by each node using the function c and a deterministic tie-breaking for nodes m_j that have same value of $c(m_j)$. ◀

Once the nodes $\mathbb{M}$ know the functions c and l, they can also compute the following functions locally.

1. We define the sequence L as the nodes $\mathbb{M}$ arranged in ascending order of values assigned by function l. The value of $Next(i)$ is j so that the node m_j is next to the node m_i in L. Similarly, the value of $Prev(i)$ is j so that the node m_j occurs just before m_i in L. We use $\perp$ to point past end or before start of L. Formally,

$$Next(i) = \begin{cases} l^{-1}(l(i)+1) & \text{if } l(i) < n-1, \\ \perp & \text{otherwise} \end{cases}$$

and

$$Prev(i) = \begin{cases} l^{-1}(l(i)-1) & \text{if } l(i) > 1, \\ \perp & \text{otherwise}. \end{cases}$$

2. Consider the sequence L. We call a node m_j a *pivot* if m_j is first node in L or $c(m_j) > c(m_{Prev(j)})$. If a node m_j is a pivot, then $F(j)$ points to itself. Otherwise, $F(j)$ points to the next pivot node in L ($\perp$ if there is no next pivot in L). Formally,

$$F(i) = \begin{cases} i & \text{if } i = l^{-1}(\min\{l(j) : c(m_j) = c(m_i)\}), \\ l^{-1}(\min\{l(j) : c(m_j) > c(m_i)\}) & \text{if } \{l(j) : c(m_j) > c(m_i)\} \neq \emptyset, \\ \perp & \text{otherwise}. \end{cases}$$

3. If m_i is a pivot node, the value of $Wait(i)$ is the number of men m_j that have $c(m_j) = c(m_i)$. Otherwise, it is zero.

$$Wait(i) = \begin{cases} |\{m_j : c(m_j) = c(m_i)\}| & \text{if } i = l^{-1}(\min\{l(j) : c(m_j) = c(m_i)\}), \\ 0 & \text{otherwise} \end{cases}$$

Our goal is to design a distributed algorithm that simulates the sequential algorithm in constant number of rounds per proposal. The broad idea of the algorithm can be explained as follows. The algorithm matches men on the left side of sequence L and simultaneously deletes elements from the preference lists of men on the right side of sequence L. As the algorithm proceeds, the left part of sequence L consisting of matched men grows. The function F is used to pass on the deletion information from the first part to the second part. The function $Wait$ is used to ensure that before a proposal is made, deletion of elements from the preference lists of men until the next pivot is finished. The functions $Next$ and $Prev$ are used to put the nodes into sequence L initially.

The precise description is in Algorithm 2 for a node m_i and in Algorithm 3 for a node w_i. The variable *counter* increments by 1 in each round. The list of unmatched men as per their order in the sequence L is maintained by the variables *next* and *prev*. The variable *match* stores the current partner ($\perp$ if none). The algorithm proceeds in *phases*. Each phase lasts eight rounds. A proposal is only sent in the first six rounds of the phase ($counter \bmod 8 < 6$) where as the deletion of women from the preference lists occurs in the last two rounds of the phase ($counter \bmod 8 \geq 6$). If a free woman receives a proposal, the initiation of her deletion from the preference lists also occurs during the first six rounds. The variable D contains the woman that should be checked for deletion in the next round and is updated during the last two rounds of the phase.

Let us now check if the distributed algorithm achieves its goal of simulating the sequential algorithm in a constant number of rounds per proposal.

8 Analysis of the Distributed Algorithm

Let us establish some helpful notations first. X_i^t is the value of the variable X stored by node m_i at the start of round $t \geq 0$ (loop at Line 4 when $counter = t$). $\widehat{X}_i^t$ is the value of the variable X stored by node w_i at the start of round t. The notation $p(t)$ represents the number of $\langle propose \rangle$ messages sent until round t starts. $DM(t)$ is the matching stored by the variables $match_i^t$ for $i \in [1, n]$ at the start of round t. Similarly, $\widehat{DM}(t)$ is the matching stored by the variables $\widehat{match}_i^t$ for $i \in [1, n]$ at the start of round t. $SM(q)$ is the matching

Algorithm 2: The algorithm executed by a node m_i to compute its match.

```
$next \leftarrow Next(i)$, $prev \leftarrow Prev(i)$;
$wait \leftarrow Wait(i)$, $match \leftarrow \bot$;
$D \leftarrow \bot$, $r \leftarrow \bot$, $accepted \leftarrow false$;
for $counter \leftarrow 0$ to $8(3\Delta + 2)n - 1$ do
    if $counter \bmod 8 = 0$ then
        if $match = \bot$ and $prev = \bot$ then
            if $wait > 0$ then
                $wait \leftarrow wait - 1$;
            else
                Send $\langle propose \rangle$ to next woman $w_j$ available in prefrence list;
    else if $counter \bmod 8 = 1$ then
        if Received $\langle accept, x \rangle$ from $w_j$ then
            $r \leftarrow x$;
            $match \leftarrow j$;
            $accepted \leftarrow true$;
    else if $counter \bmod 8 = 2$ and $accepted = true$ then
        Send $\langle SetPrev, r \rangle$ to $w_{next}$ if $next \neq \bot$;
    else if $counter \bmod 8 = 3$ then
        if Received $\langle SetPrev, r \rangle$ then
            $prev \leftarrow r$;
    else if $counter \bmod 8 = 4$ and $accepted = true$ then
        if $r = \bot$ and $next \neq \bot$ and $F(next) \neq \bot$ then
            Send $\langle SetD, (w_{match}, c(m_{Prev(next)}) + \Delta) \rangle$ to $w_{F(next)}$;
        else
            Send $\langle SetNext, next \rangle$ to $w_r$ if $r \neq \bot$;
        $next \leftarrow \bot$;
        $accepted \leftarrow false$;
    else if $counter \bmod 8 = 5$ then
        if Received $\langle SetD, x \rangle$ then
            $D \leftarrow x$;
        if Received $\langle SetNext, x \rangle$ then
            $next \leftarrow x$;
            $match \leftarrow \bot$;
    else if $counter \bmod 8 = 6$ and $D \neq \bot$ then
        Say $D = (w_j, x)$;
        Remove $w_j$ from preference list if $c(m_i) > x$;
        Send $\langle SetD, D \rangle$ to $w_{Next(i)}$ if $Next(i) \neq \bot$;
        $D \leftarrow \bot$;
    else // $counter \bmod 8 = 7$
        if Received $\langle SetD, x \rangle$ then
            $D \leftarrow x$;
```

Algorithm 3: The algorithm executed by a node w_i to compute its match.

```
1 $match \leftarrow \perp$;
2 if Received $\langle propose \rangle$ from $m_j$ then
3     if $match = \perp$ or $m_j$ ranked higher than $m_{match}$ then
4         Send $\langle accept, match \rangle$ to $m_j$ in the next round;
5         $match \leftarrow j$;
6 if Received a message $X$ other than $\langle propose \rangle$ then
7     Forward $X$ to $m_i$ in the next round;
```

computed by Algorithm 1 after q proposals are sent. Given a tuple $D = (a, b)$, $D.1 = a$ and $D.2 = b$. The notation $i(w)$ represents the value of i in Algorithm 1 when $w \in \mathbb{W}$ receives its first proposal. Consider the sequence of values $wait_i^t$ for a given t and $i \in [1, n]$ in the same order as L. $fw(t)$ is j so that $wait_j^t$ is the first positive element of the sequence ($\perp$ if all $wait_i^t \leq 0$). We define the following invariants $Inv(t)$ where $t \geq 0$ is the first round of a phase, i.e., $t = 8k$ for $k \in [0, (3\Delta + 2)n - 1]$. Note that the algorithm terminates after $8(3\Delta + 2)$ rounds. For convenience, however, we will use the start of round $e = 8(3\Delta + 2)$ to refer the invariants just after the last round.

1. At most one $\langle propose \rangle$ message is sent in a phase. If the q^{th} proposal in the sequential algorithm is sent by m to w, then the q^{th} $\langle propose \rangle$ message is sent by m to w for $q \leq p(t)$. Moreover, $DM(t) = \widehat{DM}(t) = SM(p(t))$.
2. If m_i is matched or $match_i^t \neq \perp$, then $prev_i^t = next_i^t = \perp$. Otherwise, the value $next_i^t$ points to the first unmatched node after m_i in the sequence L ($\perp$ if it does not exists). Similarly, the value $prev_i^t$ points to the last unmatched node before m_i in the sequence L ($\perp$ if it does not exists). Formally, we have the following for $match_i^t = \perp$.

$$next_i^t = \begin{cases} l^{-1}(\min\{l(j) : l(j) > l(i), match_j^t = \perp\}) & \text{if } \{l(j) : l(j) > l(i), match_j^t = \perp\} \neq \emptyset, \\ \perp & \text{otherwise} \end{cases}$$

$$prev_i^t = \begin{cases} l^{-1}(\max\{l(j) : l(j) < l(i), match_j^t = \perp\}) & \text{if } \{l(j) : l(j) < l(i), match_j^t = \perp\} \neq \emptyset, \\ \perp & \text{otherwise.} \end{cases}$$

3. If $\widehat{match}_i^t \neq \perp$, then $D_j^t.1 = w_i$ for at most one j. If such j exists, then $D_j^t.2 = i(w_i) + \Delta$. Also, m_k has deleted w_i from its preference list if $c(m_k) > i(w_i) + \Delta$ and $l(k) < l(j)$. If there is no j so that $D_j^t.1 = w_i$, then m_k has deleted w_i from its preference list if $c(m_k) > i(w_i) + \Delta$.
4. $D_i^t = \perp$ for the following nodes m_i.

$$m_i \in \begin{cases} \{m_i : l(i) \leq l(fw(t)) + Wait(fw(t)) - wait_{fw(t)}^t\} & \text{if } fw(t) \neq \perp, \\ \mathbb{M} & \text{if } fw(t) = \perp \end{cases}$$

5. Let m_i be the node so that $match_i^t = \perp$ and $prev_i^t = \perp$. If m_i sends the message $\langle propose \rangle$ in round t and $next_i^t \neq \perp$, then the nodes m_j with $l(j) \geq l(next_i^t)$ have not sent a $\langle propose \rangle$ message until round t and $wait_j^t = Wait(j)$.

For brevity, we will refer to the individual invariants as $Inv(t).1$, $Inv(t).2$ and so on. We can easily show the following assuming that the above invariants are true.

▶ **Lemma 12.** *If $Inv(t)$ holds for all $t = 8k$ where $k \geq 0$, then there is a $k \leq (3\Delta + 2)$ for which $match_i^t \neq \bot$ for all $i \in [1, n]$ and Algorithm 2, Algorithm 3 terminate in a stable matching.*

Proof. Consider the start of phase in round r when $match_j^r = \bot$ for some j. Using $Inv(r)$.2, there is a unique i for which $match_i^r = \bot$ and $prev_i^r = \bot$. Now, either $wait_i^{r+1} = wait_i^r - 1$ or m_i sends a $\langle propose \rangle$ message. As $\Sigma_{i=1}^n Wait(i) = n$, there are at most n phases where $wait_i^r > 0$. This leaves at least $(3\Delta+2)n - n = (3\Delta+1)n$ phases in which $\langle propose \rangle$ can be sent. Using $Inv(r)$.1 and Lemma 7, these are sufficient until everyone is matched in a round $y = 8k$ for $k \leq (3\Delta + 2)$.

Note that the sequential algorithm terminates as soon as all the men are matched. Using Lemma 5, $SM(p(y)) = DM(y) = \widehat{DM}(y)$ is a stable matching. The matching remains the same until termination as no further $\langle propose \rangle$ messages are sent. ◀

Now, we only need to show that $Inv(t)$ holds. We first give the following helper lemma that gives the changes during the last two rounds of the phase.

▶ **Lemma 13.** *Let $t = 8k$ for $k \in [0, (3\Delta+2)n-1]$ be the first round of a phase. If $D_j^{t+6} \neq \bot$, $Prev(j) \neq \bot$, then $D_j^{t+8} = D_{Prev(j)}^{t+6}$.*

Proof. If $D_j^{t+6} \neq \bot$, $D_k^{t+6} \neq \bot$, $Next(j) \neq \bot$ and $Next(k) \neq \bot$ for $j \neq k$, then nodes m_j, m_k send $\langle SetD, D_j^{t+6} \rangle$ and $\langle SetD, D_k^{t+6} \rangle$ respectively to $w_{Next(j)}$ and $w_{Next(k)}$ in round $t+6$. If $j \neq k$, then $Next(j) \neq Next(k)$ by definition of function $Next$. Thus, these messages are received by the nodes $w_{Next(j)}$ and $w_{Next(k)}$ which forward the message to $m_{Next(j)}$ and $m_{Next(k)}$ in round $t+7$. Upon receiving the message, the nodes $m_{Next(j)}$ and $m_{Next(k)}$ simply set their D to D_j^{t+6}, D_k^{t+6} respectively as received in the message. Thus, if m_j receives a message in round $t+7$, then $D_j^{t+8} = D_{Prev(j)}^{t+6}$. If m_j does not receive a message in round $t+7$, then $D_{Prev(j)}^{t+6} = \bot$ and $D_j^{t+8} = D_j^{t+7} = \bot$ as well. ◀

We will first show the base case of induction and later the induction step.

▶ **Lemma 14.** *$Inv(0)$ holds.*

Proof. $\underline{Inv(0).1}$ Both $DM(0)$ and $\widehat{DM}(0)$ are empty matchings by initialization which is same as $SM(p(0))$.

$\underline{Inv(0).2}$ The initialization values $Next(i)$ and $Prev(i)$ satisfy the invariant for $i \in [1, n]$.

$\underline{Inv(0).3}$ By initialization, $\widehat{match}_i^0 = \bot$ for all $i \in [1, n]$. As nodes start with complete preference lists (without any deleted entries), the invariant holds.

$\underline{Inv(0).4}$ By definition of $Wait$ we have $wait_a^0 = Wait(a) > 0$ where $a = l^{-1}(1)$. As $D_a^0 = \bot$ by initialization, the invariant follows.

$\underline{Inv(0).5}$ By initialization, $wait_a^0 = Wait(a) > 0$ where $a = l^{-1}(1)$ and propose is not sent in the first round. Thus, invariant holds as precondition is not true. ◀

▶ **Lemma 15.** *If $Inv(t)$ holds for $t = 8k$, $k \in [0, (3\Delta+2)n-1]$ and there exists $i \in [1, n]$ so that $match_i^t = \bot$, then $Inv(t+8)$ holds.*

Proof. Using $Inv(t)$.2, there is exactly one man m_i so that $match_i^t = \bot$ and $prev_i^t = \bot$. To show $Inv(t+8)$ we consider two cases: $wait_i^t \not> 0$ and $wait_i^t > 0$.

$\underline{Inv(t+8).1,\ Inv(t+8).2 \text{ when } wait_i^t \not> 0}$ As $wait_i^t \not> 0$, m_i sends a $\langle propose \rangle$ message to a woman w_q. Using $Inv(t)$.1, the set of unmatched men are the unmatched men in the matching $SM(p(t))$. Say that the $(p(t)+1)^{th}$ proposal in the sequential algorithm is sent by

m to w. Using $Inv(t).2$, $m_i = m$ and all the men m_j with $l(j) < l(i)$ are matched. Thus, the men m_j, $l(j) < l(i)$ have sent their first $\langle propose \rangle$ message in round q (say) and $wait_j^q \not> 0$. As the variable $wait$ is never incremented, $wait_j^t \not> 0$ for $l(j) < l(i)$. Using $Inv(t).4$ and the assumption $wait_i^t \not> 0$, $D_j^t = \perp$ for $l(j) \le l(i)$. Then, we can use $Inv(t).3$ to conclude that a woman w' is deleted from m_i's preference list if w' is matched and $c(m_i) > i(w') + \Delta$. Note that a woman w' is also deleted from m's preference list when m sends a proposal in the sequential algorithm if w' is matched and $c(m) > i(w') + \Delta$. Thus, we have $w = w_q$.

As no other messages are sent in round t, $\langle propose \rangle$ message is received by w_q. As in the sequential algorithm, w_q accepts the proposal if it is better. If w_q does not reply to $\langle propose \rangle$ message, then no change to variables $match$, $next$ and $prev$ are done and both $Inv(t + 8).1$, $Inv(t + 8).2$ hold. If w_q replies with $\langle accept, x \rangle$ message, then we have the following cases depending on the values of x and $next_i^t$.

Case (a): $x = \perp$, $next_i^t = \perp$ The changes to the variables $match$, $next$ and $prev$ until the end of the phase are as follows.

1. $\widehat{match}_q^{t+2} = i$ as w_q updates its $match$ variable in round $t + 1$.
2. $match_i^{t+2} = q$ as m_i receives $\langle accept, x \rangle$ in round $t + 1$.
3. $next_i^{t+5} = \perp$ as m_i sets the variable in round $t + 4$.

Thus,

1. $\widehat{match}_q^{t+8} = i$,
2. $match_i^{t+8} = q \neq \perp$,
3. $next_i^{t+8} = \perp$, and
4. $prev_i^{t+8} = \perp$ as $prev_i^t = \perp$ by assumption.

So $Inv(t + 8).1$, $Inv(t + 8).2$ hold in this case.

Case (b): $x = \perp$, $next_i^t \neq \perp$ The changes to the variables $match$, $next$ and $prev$ until the end of the phase are as follows.

1. $\widehat{match}_q^{t+2} = i$ as w_q updates its $match$ variable in round $t + 1$.
2. $match_i^{t+2} = q$ as m_i receives $\langle accept, x \rangle$ in round $t + 1$.
3. $prev_a^{t+4} = \perp$ where $a = next_i^t$ as m_i sends $\langle SetPrev, \perp \rangle$ to w_a in round $t + 2$ which forwards it to m_a in round $t + 3$.
4. $next_i^{t+5} = \perp$ as m_i sets the variable in round $t + 4$.

Thus,

1. $\widehat{match}_q^{t+8} = i$,
2. $match_i^{t+8} = q \neq \perp$,
3. $next_i^{t+8} = \perp$,
4. $prev_i^{t+8} = \perp$ as $prev_i^t = \perp$ by assumption, and
5. $prev_a^{t+8} = \perp$ where $a = next_i^t$.

So $Inv(t + 8).1$, $Inv(t + 8).2$ hold in this case.

Case (c): $x \neq \perp$, $next_i^t = \perp$ The changes to the variables $match$, $next$ and $prev$ until the end of the phase are as follows.

1. $\widehat{match}_q^{t+2} = i$ as w_q updates its $match$ variable in round $t + 1$.
2. $match_i^{t+2} = q$ as m_i receives $\langle accept, x \rangle$ in round $t + 1$.
3. $next_i^{t+5} = \perp$ as m_i sets the variable in round $t + 4$.

4. $\mathit{next}_r^{t+6} = \perp$ and $\mathit{match}_r^{t+6} = \perp$, where $r = x$ as m_i sends $\langle \mathit{SetNext}, \perp \rangle$ to w_r in round $t+4$ which forwards it to m_r in round $t+5$.

Thus, we have the following for $r = x$.

1. $\widehat{\mathit{match}}_q^{t+8} = i$,
2. $\mathit{match}_i^{t+8} = q \neq \perp$, $\mathit{next}_i^{t+8} = \perp$, $\mathit{prev}_i^{t+8} = \perp$ as before,
3. $\mathit{next}_r^{t+8} = \perp$, $\mathit{match}_r^{t+8} = \perp$, and
4. $\mathit{prev}_r^{t+8} = \perp$ as $\mathit{match}_r^t \neq \perp$ by assumption, so $\mathit{prev}_r^t = \perp$ by $\mathit{Inv}(t).2$.

As before, $\mathit{Inv}(t+8).1$ holds in this case as well. As $\mathit{next}_i^t = \perp$, m_r is the only unmatched man left and $\mathit{Inv}(t+8).2$ also holds.

Case (d): $x \neq \perp$, $\mathit{next}_i^t \neq \perp$ The changes to the variables *match*, *next* and *prev* until the end of the phase are as follows.

1. $\widehat{\mathit{match}}_q^{t+2} = i$ as w_q updates its *match* variable in round $t+1$.
2. $\mathit{match}_i^{t+2} = q$ as m_i receives $\langle \mathit{accept}, x \rangle$ in round $t+1$.
3. $\mathit{prev}_a^{t+4} = r$ where $a = \mathit{next}_i^t$ and $r = x$ as m_i sends $\langle \mathit{SetPrev}, r \rangle$ to w_a in round $t+2$ which forwards it to m_a in round $t+3$.
4. $\mathit{next}_i^{t+5} = \perp$ as m_i sets the variable in round $t+4$.
5. $\mathit{next}_r^{t+6} = a$ and $\mathit{match}_r^{t+6} = \perp$, where $a = \mathit{next}_i^t$ and $r = x$ as m_i sends $\langle \mathit{SetNext}, a \rangle$ to w_r in round $t+4$ which forwards it to m_r in round $t+5$.

Thus, we have the following for $r = x$ and $a = \mathit{next}_i^t$.

1. $\widehat{\mathit{match}}_q^{t+8} = i$,
2. $\mathit{match}_i^{t+8} = q \neq \perp$, $\mathit{next}_i^{t+8} = \perp$, $\mathit{prev}_i^{t+8} = \perp$ as before,
3. $\mathit{prev}_r^{t+8} = \perp$, $\mathit{match}_r^{t+8} = \perp$ as before, and
4. $\mathit{next}_r^{t+8} = a$.

Using $\mathit{Inv}(t).5$, $l(r) < l(a)$ and these changes are sufficient for $\mathit{Inv}(t+8).2$ to hold. Also, $\mathit{Inv}(t+8).1$ holds as well.

$\mathit{Inv}(t+8).3$ when $\mathit{wait}_i^t \not> 0$ The invariant needs to be checked only when the value of variable D changes. This only happens upon receiving the message $\langle \mathit{SetD}, x \rangle$.

If the node m_i receives $\langle \mathit{accept}, y \rangle$ in round $t+1$ with $y \neq \perp$, then $\langle \mathit{SetD}, x \rangle$ is only sent in round $t+6$ by a node m_j if $D_j^{t+6} \neq \perp$. Using Lemma 13, if $\widehat{\mathit{match}}_a^t \neq \perp$, $D_l^t.1 = w_a$ and $\mathit{Next}(l) \neq \perp$, then $D_{\mathit{Next}(l)}^{t+8}.1 = w_a$. Using $\mathit{Inv}(t).3$, we only need to check deletion of w_a from m_l's list. Before m_l sends $\langle \mathit{SetD}, x \rangle$, it deletes $D_l^t.1 = w_a$ if $c(m_l) > D_l^t.2$. Using $\mathit{Inv}(t).3$, we have $D_l^t.2 = i(w_a) + \Delta$ and $\mathit{Inv}(t+8).3$ holds.

If the node m_i receives $\langle \mathit{accept}, \perp \rangle$ in round $t+1$ from w_q, then $\langle \mathit{SetD}, x \rangle$ is only sent in rounds $t+4$ and $t+6$. In round $t+4$, m_i sends $\langle \mathit{SetD}, x \rangle$ to $w_{F(a)}$ where $a = \mathit{next}_i^{t+4}$ and $x = (w_q, c(m_{\mathit{Prev}(a)}) + \Delta)$. Using $\mathit{Inv}(t).5$, all nodes m_j with $l(j) \geq l(a)$ never sent a $\langle \mathit{propose} \rangle$ message before and all the other nodes m_k already sent a $\langle \mathit{propose} \rangle$ message as they are matched. Using $\mathit{Inv}(t).1$ and $\mathit{Inv}(t+8).1$, $i(w_q) = \max\{c(m_j) : m_j \text{ sent } \langle \mathit{propose} \rangle \text{ until round } (t{+}4)\} = c(m_{\mathit{Prev}(a)})$. If m_a is a pivot node, then $F(a) = a$ by definition and $c(m_{F(a)}) = \min\{c(m_j) : c(m_j) > i(w_q)\} \leq \min\{c(m_j) : c(m_j) + \Delta > i(w_q)\}$. If m_a is not a pivot node, then $c(m_a) = c(m_{\mathit{Prev}(a)}) = i(w_q)$ and again $c(m_{F(a)}) = \min\{c(m_j) : c(m_j) > i(w_q)\} \leq \min\{c(m_j) : c(m_j) + \Delta > i(w_q)\}$. Therefore, $l(p) \geq l(F(a))$ if $c(m_p) > i(w_q) + \Delta$. Now we use Lemma 13 and $\mathit{Inv}(t).3$ as before to conclude that $\mathit{Inv}(t+8).3$ holds after $\langle \mathit{SetD}, x \rangle$ is sent in round $t+6$.

$\underline{Inv(t+8).4 \text{ when } wait_i^t \not> 0}$ If m_a for $a = next_i^t \neq \perp$ is a pivot node, then $F(a) = a$ and $Wait(a) > 0$ using definition of *Wait*. Using $Inv(t).5$, we conclude $wait_a^t = Wait(a) > 0$. As all nodes m_j with $l(j) < l(a)$ sent a $\langle propose \rangle$ message at least once, $wait_j^t \not> 0$. Thus, $a = F(a) = fw(t)$ if m_a is a pivot node. If m_a for $a = next_i^t \neq \perp$ is not a pivot node, then $Wait(k) = 0$ for $l(a) \leq l(k) < l(F(a))$ using definition of *Wait*. As before $wait_j^t \not> 0$ for all nodes m_j with $l(j) < l(a)$. Using $Inv(t).5$, we conclude that $wait_{F(a)}^t = Wait(F(a)) > 0$. Thus, we conclude that $F(a) = fw(t)$ irrespective of whether m_a is a pivot node or not.

Using $Inv(t).4$, we conclude $D_j^t = \perp$ for $l(j) \leq l(a)$. As *wait* is not changed in this phase, $Inv(t+8).4$ holds if $D_j^{t+8} = \perp$ for $l(j) \leq l(a)$. In round $t+4$, m_i may send $\langle SetD, x \rangle$ to w_a which then forwards it to m_a in round $t+5$ so $D_a^{t+6} \neq \perp$. Using Lemma 13, we again have $D_j^{t+8} = \perp$ for $l(j) \leq l(a)$.

If $next_i^t = \perp$, then all nodes m_k, $k \in [1, n]$ already sent a $\langle propose \rangle$ message at least once. Thus, $wait_k^t \not> 0$. Using $Inv(t).4$, $D_k^t = \perp$. As no $\langle SetD, x \rangle$ message is sent until round $t+8$ starts, $D_k^{t+8} = \perp$ as well.

$\underline{Inv(t+8).5 \text{ when } wait_i^t \not> 0}$ Using $Inv(t).5$, the nodes m_j with $l(j) \geq l(a)$ have not sent a $\langle propose \rangle$ message until round t for $a = next_i^t \neq \perp$. Thus, if a node m_k is matched, then $l(k) < l(a)$.

Now, if m_i receives $\langle accept, x \rangle$ in round $t+1$ with $x \neq \perp$, then $match_r^{t+8} = \perp$ for some r so that $l(r) < l(a)$. Using $Inv(t+8).2$, only m_r is the node that satisfies $match_r^{t+8} = \perp$ and $prev_r^{t+8} = \perp$. Using $Inv(t+8).2$, we also conclude that $next_r^{t+8} = a$. The nodes m_j with $l(j) \geq l(a)$ do not send a $\langle propose \rangle$ message in this phase nor change their *wait* variables. Using $Inv(t).5$, the nodes m_j with $l(j) \geq l(a)$ satisfy $wait_j^t = Wait(j)$. Thus, $Inv(t+8).5$ holds if a node m_r is rejected in this phase.

If m_i receives $\langle accept, x \rangle$ in round $t+1$ with $x = \perp$, then using $Inv(t+8).2$ we conclude that m_a is the only node that satisfies $match_a^{t+8} = \perp$ and $prev_a^{t+8} = \perp$. The nodes m_j with $l(j) \geq l(a)$ do not change their *match* variables in this phase, do not send a $\langle propose \rangle$ message in this phase and do not change their *wait* variables in this phase. Using $Inv(t).5$, the nodes m_j with $l(j) \geq l(a)$ satisfy $wait_j^{t+8} = Wait(j)$. In round $t+8$, only m_a may send a $\langle propose \rangle$ message Thus, if $next_a^{t+8} \neq \perp$, then the nodes m_j with $l(j) \geq l(next_a^{t+8})$ have not sent a $\langle propose \rangle$ message until round $t+8$ and have $wait_j^{t+8} = Wait(j)$.

If m_i does not receive a message from w_q in round $t+1$, then none of the nodes change their *match*, *prev*, *next* or *wait* variables. Also, only m_i may send a $\langle propose \rangle$ message in round $t+8$. Using $Inv(t).5$, the invariant also follows when m_i does not receive a message from w_q in round $t+1$.

$\underline{Inv(t+8) \text{ when } wait_i^t > 0}$ In this case, m_i decrements its *wait* variable instead of sending a $\langle propose \rangle$ message. As a result, for all nodes m_j we have $next_j^{t+8} = next_j^t$, $prev_j^{t+8} = prev_j^t$ and $match_j^{t+8} = match_j^t$. Thus, there is nothing to check for $Inv(t+8).1$ and $Inv(t+8).2$. Using Lemma 13, $Inv(t+8).4$ holds as well because $wait_i^{t+8} = wait_i^t - 1$. Verifying $Inv(t+8).3$ in this case is same as verifying it when the $\langle propose \rangle$ message sent by m_i is replied with an $\langle accept, x \rangle$ where $x \neq \perp$.

We consider two cases to check for $Inv(t+8).5$. First: if $wait_i^t > 1$, then $wait_i^{t+8} > 0$. Thus, the message $\langle propose \rangle$ will not be sent in round $t+8$ and nothing needs to be checked. Second: if $wait_i^t = 1$, then $wait_i^{t+8} = 0$. Thus, the node m_i never sent a $\langle propose \rangle$ message until round t as otherwise $wait_i^t \not> 0$. Also, no node m_j with $l(j) \geq l(i)$ sent a $\langle propose \rangle$ message as otherwise m_i must have sent $\langle propose \rangle$ message at least once. Thus, no m_j with $l(j) > l(i)$ ever got matched and $prev_j^{t'} \neq \perp$ for the first round t' of any phase where $t' \leq t$. Thus, $wait_j^{t+8} = Wait(j)$ for $l(j) > l(i)$. As m_i sends $\langle propose \rangle$ in round $t+8$, we conclude that $Inv(t+8).5$ holds. ◀

▶ **Theorem 16.** *Algorithm 2 and Algorithm 3 compute a stable matching where each node needs $O(n(\Delta + 1))$ messages of size $O(\log n)$.*

Proof. Using Lemma 14, Lemma 15 and Lemma 12, each node computes its partner in a stable matching in $O(n(\Delta + 1))$ rounds. As a node receives at most one message of size $O(\log n)$ per round, the statement follows. ◀

9 Conclusion and Open Problems

We considered the distributed version of stable matching where preference lists of one set of participants are almost similar. Given the non-negative similarity parameter $\Delta \leq n - 1$, any node can compute its partner in a stable matching by receiving $O(n(\Delta + 1))$ values. Also, there is always a node that must receive $\Omega(n/\log n)$ values.

It still remains to find out if the above algorithm is optimal (up to a logarithmic factor) if Δ is not constant. Furthermore, it may also be interesting to have an algorithm that uses $O(1)$ sized messages instead of $O(\log n)$ sized messages.

References

1 Nir Amira, Ran Giladi, and Zvi Lotker. Distributed Weighted Stable Marriage Problem. In *17th International Colloquium on Structural Information and Communication Complexity (SIROCCO), Sirince, Turkey*, June 2010.

2 Ismel Brito and Pedro Meseguer. Distributed Stable Matching Problems. In *11th International Conference on Principles and Practice of Constraint Programming (CP), Sitges, Spain*, October 2005.

3 Ismel Brito and Pedro Meseguer. Distributed stable matching problems with ties and incomplete lists. In *12th International Conference on Principles and Practice of Constraint Programming (CP), Nantes, France*, September 2006.

4 Patrik Floréen, Petteri Kaski, Valentin Polishchuk, and Jukka Suomela. Almost Stable Matchings by Truncating the Gale–Shapley Algorithm. *Algorithmica*, September 2010.

5 D. Gale and L. S. Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*, January 1962.

6 Dan Gusfield and Robert W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. The MIT Press, 1989.

7 Robert W. Irving, David F. Manlove, and Sandy Scott. The Stable Marriage Problem with Master Preference Lists. *Discrete Applied Mathematics*, August 2008.

8 Deepak Kapur and Mukkai S. Krishnamoorthy. Worst-case Choice for the Stable Marriage Problem. *Information Processing Letters*, July 1985.

9 Alex Kipnis and Boaz Patt-Shamir. A Note on Distributed Stable Matching. In *29th IEEE International Conference on Distributed Computing Systems (ICDCS), Montreal, QC, Canada*, June 2009.

10 Eyal Kushilevitz. Communication Complexity. In *Advances in Computers*, volume 44. Elsevier, 1997.

11 Rafail Ostrovsky and Will Rosenbaum. Fast Distributed Almost Stable Matchings. In *34th ACM Symposium on Principles of Distributed Computing (PODC), Donostia-San Sebastián, Spain*, July 2015.

12 Michael J. Quinn. A Note on Two Parallel Algorithms to Solve the Stable Marriage Problem. *BIT Numerical Mathematics*, September 1985.

13 S. S. Tseng and R. C. T. Lee. A Parallel Algorithm to Solve the Stable Marriage Problem. *BIT Numerical Mathematics*, September 1984.